Multicommodity Facility Location under Group Steiner Access Cost

Laura J. Poplawski* Rajmohan Rajaraman†

Abstract

Motivated by publish-subscribe mechanisms in networks, we introduce a new class of multicommodity facility location problems: Multicommodity Group Steiner Facility Location (MGSFL). The input to MGSFL consists of a metric space over a given set of locations, a cost function which provides a building cost for each commodity at each location, a set of clients located at various points in the metric, and the set of commodities that each client is interested in reaching. A solution to MGSFL consists of (a) for each commodity, the locations where facilities are built, and (b) for each client, a tree connecting the client to at least one facility for each commodity in its interest set. The goal is to minimize the sum of the total facility building costs and the metric cost of the client trees.

MGSFL is a natural generalization of the well-studied Group Steiner Tree problem, which is equivalent to the special case of MGSFL in which every building cost is either 0 or ∞ and there is only one client. We also note that given the facility locations, the best client tree is an optimal solution to an appropriate Group Steiner Tree instance. Since the Group Steiner Tree problem is hard to approximate to within a factor of $\Omega(\log^{2-\epsilon} m)$ times optimum unless **NP** has quasi-polynomial Las Vegas algorithms, where m is the number of commodities, the same hardness result immediately extends to MGSFL.

Our main result is a randomized $2^{O(\sqrt{\log n}\log\log n)}$-approximation algorithm for MGSFL, where n is the number of clients. We also present deterministic poly-logarithmic approximations for three special cases. We give an $O(\log n)$-approximation algorithm when the facility building costs differ only by commodity, not by location. We present an $O(\log^4 n \log m)$-approximation algorithm when the interest sets are laminar — i.e., for each pair of clients, either their interest sets do not intersect or else one client's interest set is contained within the other client's interest set. We end with an $O(\log n)$-approximation algorithm when there are no building costs but each commodity must be built exactly once.

1 Introduction

Facility location is one of the oldest and most well-studied algorithmic problems, applicable in widely diverse areas such as business operations, network management, and graph theory. Most existing multicommodity facility location variants apply to cases in which the clients are physically moving between the facilities to obtain the various commodities. We are concerned instead with a situation in which clients simply want to *connect* to facilities serving various commodities. This is applicable, for instance, in the case of a publish-subscribe network in which clients are subscribers in the network and commodities are content sources to be published. Each subscriber (a node in the network) is willing to pay for only the least expensive set of edges required to connect it to a source for each item of relevant published content. Content distribution networks (CDNs) give another real-world application: the CDN controls a set of links and wants to minimize both the cost of creating data caches and the link usage by its end users. We can estimate the link usage by an end user as the sum of the costs of the links necessary to connect the user to its required caches.

These network-design applications lead to a fairly concrete problem definition which is a variation of multicommodity facility location. We are given a set of nodes (or clients) located in a network, plus a set of content types (or commodities) that must be placed (a facility must be built) at nodes in the network. Each client is interested in some subset of the commodities, which we refer to as the *interest set* of the client. The *interest graph* is the bipartite graph of clients and commodities with an edge from each client to the commodities in its interest set. We are given the cost for each commodity to build a facility at each location (which may differ by commodity and by location). We must pay separately to build each commodity, even if multiple commodities are built at the same location. Each client will choose the minimum cost set of edges that connect it to some facility for each commodity in its interest set. In other words, each client will choose the minimum Group Steiner Tree [18] that connects it to its interest set. Our goal is an algorithm to decide where to build each commodity to minimize the total building cost plus the total Group Steiner Tree costs for the clients. We call this the *Multicommodity Group Steiner Facility Location* problem (MGSFL).

From a theoretical perspective, MGSFL is particularly interesting as a generalization of two distinct **NP**-complete problems: Facility Location (equivalent to MGSFL with one commodity) and Group Steiner Tree (equivalent to MGSFL with a single client and with each building cost set to either 0 or ∞). It is

*Raytheon BBN Technologies.
†Northeastern University.

already known that Group Steiner Tree cannot be approximated to within a factor of $\Omega(\log^{2-\epsilon} m)$ (where m is the number of commodities), even on trees, unless **NP** has quasi-polynomial Las Vegas algorithms [11]. Any approximation algorithm for MGSFL will also give an algorithm for solving Group Steiner Tree.

1.1 Our Results Because we are working within an $\Omega(\log^{2-\epsilon} m)$ bound on the hardness of approximating MGSFL (where m is the number of commodities), we concentrate on a poly-logarithmic approximation ratio. Our approach is to attempt a poly-logarithmic approximation ratio on trees, and then apply the results from [7] to get an approximation algorithm for general metrics. In the following, we use n to represent the number of clients and m to represent the number of commodities.

- **Building costs vary by commodity, not by location**. (Section 3) We consider MGSFL on trees when the cost of building each commodity is the same regardless of where the facility is built. For this special case, which we show is APX-hard, we give a 9-approximation algorithm, implying an $O(\log n)$ approximation for general metrics.
- **General** MGSFL. (Section 4) Our main result is a randomized $2^{O(\sqrt{\log n}\log\log n)}$-approximation algorithm for general MGSFL. Our algorithm is a natural generalization of the Group Steiner Tree algorithm of Garg, Konjevod, and Ravi [8] using a dependent rounding method, whose analysis may be of independent interest.

Since we were unable to obtain a poly-logarithmic approximation ratio for the general problem, we also examine a version of MGSFL with specialized interest sets. Content and information sources are often organized hierarchically; consequently, client interests can often be represented together as a laminar graph. We consider the special case where the interest sets of the clients are laminar: given any two clients, either the interest sets do not intersect or else one client's interest set is contained within the other client's interest set. This version still has the same approximation hardness bound, but it has nice properties that make the problem more approachable.

- **Laminar clients**. (Section 5) We give an $O(\log^3 n \log m)$-approximation algorithm for tree metrics when the interest graph is laminar by client, implying an $O(\log^4 n \log m)$ approximation for general metrics.

Often, k-median problems go hand-in-hand with facility location problems, since this is the same problem restricted to a certain number of buildings per commodity (instead of allowing unlimited buildings with a cost per building). We consider only the 1-median version.

- **1-Median**. (Section 6) We give an optimal deterministic algorithm for the 1-median variant of MGSFL on tree metrics with no building costs, implying an $O(\log n)$ approximation for general metrics. This is the equivalent to the problem with building costs varying only by commodity if all building costs are very high compared to the edge lengths.

We conclude with some open problems in Section 7.

1.2 Related Work Variants of Facility Location have been studied for years. Single commodity versions include uncapacitated ([19], [9], [14], [13], [15]), capacitated ([6], [16]), and k-median, in which at most k facilities can be built ([4], [1]). In [17], the first well-known multicommodity version, multiple commodities may be built at the same facility with decreasing marginal cost. [17] gives an $O(\log t)$ approximation algorithm and a matching hardness result, where t is the maximum number of commodities allowed at each facility. The results from [17] apply to the k-median version as well.

A Facility Location variant closely related to MGSFL — Group Facility Location — is discussed in [12]. In Group Facility Location, the goal is to open a set of facilities for each commodity, and each commodity will build a Steiner forest (a set of trees rooted at the open facilities) connecting itself to all interested clients. The goal is to minimize the sum over all commodities of (1) the building costs plus (2) the cost of the Steiner forest for the commodity. This is different from MGSFL because the *commodities* are building Steiner forests rather than the clients. Since a Group Facility Location algorithm chooses the locations for the commodities, building forests around the commodities allows the algorithm to choose the roots of the forests. In MGSFL, however, the roots are fixed and an algorithm determines the leaves, yielding a different problem from a technical standpoint.

Since each of our clients is solving the Group Steiner Tree problem, introduced in [18], our results also build on previous work related to Group Steiner Tree. [8] gives a randomized $O(\log n \log m)$-approximation algorithm for Group Steiner Tree on a tree, where n is the number of nodes and m is the number of groups. [3] derandomizes the algorithm from [8]. [10] gives an $\Omega(\log^2 m)$ lower bound on the integrality ratio of Group Steiner Tree, even on trees, and this is extended to an $\Omega(\log^{2-\epsilon} m)$ approximation lower bound in [11]. [5] provides a combinatorial algorithm with a poly-

logarithmic approximation ratio.

MGSFL is also a generalization of the multicast "push-pull" data dissemination problem introduced in [2]. In fact, one problem discussed in [2] is precisely a 1-Median version of MGSFL in which exactly two clients are interested in each commodity. [2] also considers a number of other problem variations that differ significantly from MGSFL.

2 Formal Definition

An instance of MGSFL is defined by a tuple $\langle V, E, M, I, d, c\rangle$, in which V is the set of nodes, E is the set of edges, M is the set of commodities, $I : V \to \mathbb{Z}$ is the interest set function, $d : E \to \mathbb{Z}$ is the length function, and $c : M \times V \to \mathbb{Z}$ is the building cost function. The interest set of v, $I(v)$, represents the set of commodities in which node v is interested. The length of edge $e = (u, v)$ is $d(e)$ or $d(u, v)$. The cost of building a facility for commodity t at node v is $c(t, v)$. We will use $n = |V|$ and $m = |M|$ for notational convenience. We refer to any node with a nonempty interest set as a *client*. Our formulation allows at most one client at any node. This is without loss of generality since multiple clients at a node can be modeled by simply creating a new node for each client connected by 0 length edges.

A solution to an MGSFL instance consists of a collection of location sets $\{L_t : t \in M\}$ and a collection of trees $\{S_v : v \in V\}$; L_t specifies the nodes at which a facility for commodity t will be built, and S_v connects v to at least one node in L_t for each $t \in I(v)$. When the tree S_v successfully includes a node at which a facility for $t \in I(v)$ has been built, we say that client v *meets* commodity t. We extend the length function d and define $d(S_v)$ to be the sum of lengths of the edges in S_v. An optimal solution to the MGSFL instance minimizes the following.

$$\sum_{v \in V} d(S_v) + \sum_{t \in M} \sum_{v \in L_t} c(t, v)$$

In the special case where there is only one client and each building cost is set to be either 0 or ∞, MGSFL is equivalent to the well studied Group Steiner Tree problem [18, 8, 3, 10, 5], in which we are given m groups of nodes and want to find the minimum cost tree connecting a given root node to at least one node from each of these groups. It is known that even when the underlying graph is a tree, Group Steiner Tree cannot be approximated to within $\Omega(\log^{2-\epsilon} m)$ (where m is the number of groups) unless **NP** has quasi polynomial Las-Vegas algorithms [11]. Thus, our focus in this paper is on poly-logarithmic approximations for MGSFL. We use a standard approach toward this goal: we concentrate on solving MGSFL on trees, and invoke metric embedding machinery [7] to derive results for general metrics. From [7], there is a randomized embedding from any metric into a tree with maximum distortion $O(\log n)$ (where n is the number of nodes in the metric), so any solution on a tree with approximation ratio R will achieve approximation ratio $O(R \log n)$ on an arbitrary metric.

Henceforth, we concentrate on algorithms for MGSFL when the underlying graph is a tree $T = (V, E)$. We add the following additional notation. Let $h(T)$ be the height of a tree T, P_{uv} be the unique path connecting nodes u and v in T, a_{uv} be the least common ancestor of nodes u and v in T, T_e be the subtree of T rooted at and including edge e, and T_u be the subtree of T rooted at node u. We also assume that all clients are located at the leaves, and that all commodities must be built at leaves. This does not restrict our problem, since we can always add an edge of length 0 to each node in the graph, effectively converting our non-leaf nodes into leaves.

For the benefit of the reader, Table 1 summaries the above notation.

$T = (V, E)$	A tree consisting of nodes (V) and edges (E)
n	The number of nodes ($\|V\|$).
T_e, T_u	The subtree of T rooted at edge e, node u.
$h(T)$	The height of tree T.
P_{uv}	The set of edges along the unique path in T from node u to node v.
a_{uv}	The least common ancestor of nodes u and v in T.
$d(e)$, $d(u, v)$	The length of edge $e = (u, v)$.
M, m	The set of all commodities (M) and the number of commodities (m).
$I(v)$	The subset of commodities in which the client at node v is interested.
$c(t, v)$	The cost of building a facility for commodity t at node v.
S_v	The group Steiner tree used for client v to connect it to all commodities $t \in I(v)$.

Table 1: Notation

Linear Program for MGSFL. Many of our algorithms use the following linear program, which gives a fractional solution to MGSFL on a tree, assuming without loss of generality that we may only build commodities at the leaves.

In the LP, the variable y_{tu} represents the amount commodity t is built at node u, x_{jtu} represents the amount client j meets commodity t at node i, and z_{je} represents the amount client j crosses edge e.

minimize

$$\sum_{j,e} z_{je} \cdot d(e) + \sum_{u,t} y_{tu} \cdot c(u,t)$$

subject to

(2.1)
$$z_{je} \geq \sum_{u:e \in P_{ju}} x_{jtu} \qquad \text{for all } j,t,e$$

(2.2)
$$\sum_{u} x_{jtu} \geq 1 \qquad \text{for all } j, t \in I(j)$$

(2.3)
$$y_{tu} \geq x_{jtu} \qquad \text{for all } j,t,u$$
$$y_{tu} = 0 \qquad \text{for all } t, \text{ for all } u \text{ not at a leaf}$$
$$y_{tu} \geq 0 \qquad \text{for all } t,u$$
$$x_{jtu} \geq 0 \qquad \text{for all } j,t,u$$
$$z_{je} \geq 0 \qquad \text{for all } j,e$$

In addition, we want to set the following, which will be used in the algorithms. Notice that w_{jte} and w_{jtv} are each at most 1 for any j,t,e,v.

- $w_{jtu} = \sum_{v \in T_u} x_{jtv}$
- $w_{jte} = \sum_{v \in T_e} x_{jtv}$
- $Y_{tu} = \sum_{v \in T_u} y_{tv}$

Since the LP gives a fractional solution to MGSFL, we may describe the fractional solution with such phrases as "client j meets commodity t at least $\frac{1}{4}$ in subtree T_e" or "commodity t is built at most $\frac{1}{2}$ at node v." A fractional meeting of amount δ between j and t at v means that $\min_{e \in P_{jv}} z_{je} \geq \delta$ and $y_{tv} \geq \delta$. A fractional building amount δ for t at v means $y_{tv} \geq \delta$.

It is worth noting that the same linear program, appropriately specialized, has been used to solve the Group Steiner Tree problem [8, 3, 10, 5]. Therefore the integrality ratio of this LP is at least $\Omega(\log^2 m)$ in the general case, as proved in [10].

We close this section with the following Theorem proved in [5], and an immediate Corollary, which we use later in this paper. For a given tree T, let $h(T)$ denote the height of T and let $d_T(u,v)$ represent the distance between nodes u and v in T.

THEOREM 2.1. Chekuri, Even, Kortsarz [5, Section 4] *Then, we can find another tree T' such that $h(T') \leq 3 \cdot \log_\alpha n$, and for any u, v, $d_T(u,v) \leq d_{T'}(u,v) \leq \alpha \cdot d_T(u,v)$.*

COROLLARY 2.1. *Given a tree T with n nodes, we can find a tree T' with $h(T') \leq O(\sqrt{\log_2 n})$, and for any u, v, $d_T(u,v) \leq d_{T'}(u,v) \leq O(2^{\sqrt{\log_2 n}}) \cdot d_T(u,v)$.*

3 Building costs vary by commodity, not by location

In this section, we restrict the building cost function so that $c(t,v) = c(t,v')$ for all $t,v,v' \neq v$. For this special case, we present a polynomial-time $O(\log n)$-approximation algorithm. We achieve this by developing a 9-approximation algorithm for tree metrics, and then invoking the metric embedding machinery. Before we present our algorithm for the tree, we show that MGSFL problem is APX-hard, even for the special case when all building costs are uniform and the underlying network is a tree. This implies, our approximation for trees is within a constant factor of the best achievable unless $\mathbf{P} \neq \mathbf{NP}$.

THEOREM 3.1. *When building costs are uniform,* MGSFL *is APX-hard even for tree networks.*

Proof. The proof is by reduction from set cover. Let $\mathcal{S}$ be a set cover instance with U being the universe of n elements, and S being the collection of sets. We assume furthermore that each set consists of at most B elements, and the number of sets an element belongs to is at most B. We construct an MGSFL instance $\langle V, E, M, I, d, c \rangle$ as follows. The set V equals $S \cup \{r\}$ where r is a special root node. The set E of edges equals $\{(r,s) : s \in S\}$. We have a commodity for each element, so M equals U. The length of every edge is 1 and the cost of building any commodity at any location is C ($C > 1$). At every leaf node s, for every commodity $t \in s$, we have k clients each with interest set equal to $\{s\}$, where k is chosen to be larger than $2Cn^2$. Finally, we have one client at the root with interest set equal to U.

We now consider solutions to the MGSFL instance. One solution X is to build a facility for commodity t at leaf s whenever t is in s, and for the client tree of the client at root r to be formed by connecting the root to the leaf nodes in a subset S' of S that forms the minimum set cover solution to $\mathcal{S}$. The cost of X equals $\text{opt}(\mathcal{S}) + C \sum_{s \in S} |s|$, where $\text{opt}(\mathcal{S})$ is the cost of a minimum set cover. $\text{opt}(\mathcal{S}) < n$, since the n nodes include $|S|$ leaves plus one root, while $\text{opt}(\mathcal{S})$ includes at most all $|S|$ sets. $C \sum_{s \in S} |s| \leq C|S|B < Cn^2$.

Consider any solution Y. If there exists a leaf node s and a commodity $t \in s$ such that no facility for t is built at s, then the cost incurred by the k clients to meet commodity t is at least k. Since $k > 2Cn^2$ (which exceeds the cost of X), Y is optimal only if at every leaf

s a facility is built for every commodity in s; otherwise, we can replace Y by another solution that has smaller cost and satisfies this property. Furthermore, we can assume without loss of generality that no facility is built at the root since any such facility can be removed, and at most one edge added to the root client tree while saving at least $C - 1 > 0$ cost. Thus, the set T of leaves where the root client meets the commodities in $U \setminus R$ is a valid set cover solution. The cost of Y equals $C \sum_{s \in S} |s| + \text{cost}(T)$, where $\text{cost}(T)$ is the cost of the set cover solution T.

Suppose we obtain an α-approximation to the MGSFL instance. Then, we obtain a solution to the set cover instance of cost at most $(\alpha - 1)C \sum_{s \in S} |s| + \alpha \text{opt}(\mathcal{S})$, which is at most $(\alpha - 1)CBn + \alpha \text{opt}(\mathcal{S})$. Since $\text{opt}(\mathcal{S})$ is at least n/B, we obtain that we have a solution to the set cover instance with cost at most $((\alpha - 1)CB^2 + \alpha)\text{opt}(\mathcal{S})$. Since C and B are constants, we can set $\alpha + (\alpha - 1)CB^2$ to be a constant strictly bigger than 1 by setting α sufficiently close to 1. Given that the set cover problem, even with the restriction of bounded set sizes and bounded number of occurrences for each element, is APX-hard, it follows that the MGSFL problem with the restriction that the underlying network is a tree and building costs are uniform, is APX-hard.

We next present our algorithm. We round a solution to the LP using Algorithm 1. Algorithm 1 builds three "types" of facilities. Examining one commodity at a time, we skim down the tree in a depth-first search order, starting at the root. We built the first type of commodity, Type A, at some node u if there are two subtrees rooted at children of u such that the LP built the commodity at least $\frac{1}{3}$ in each of these subtrees. As we skim down the tree, each time we've accumulated another unused $\frac{1}{3}$ of built facility for the commodity, we build a Type B facility. If we hit a point where no subtree has at least $\frac{2}{3}$ total building for the commodity, then we build a facility of Type C and stop building this commodity down this path.

Algorithm 1 Approximation algorithm for the version of MGSFL with no budget and with building costs varying only by commodity, not by location.

1: Solve the LP.
2: **for** each commodity t **do**
3: **for** each node r_0 in the tree, in a depth-first-search order **do**
4: Let $(r_0, r_1, r_2, \ldots, r_f)$ = the nodes along the path from r_0 to the root of the tree (r_f).
5: **if** $Y_{tr_0} \geq \frac{1}{3}$ **then**
6: **if** there are no "Type C" buildings for commodity t along path $(r_1, r_2, \ldots, r_f)$ **then**
7: **if** there are two children c_1 and c_2 of r_0 such that $Y_{tc_1} \geq \frac{1}{3}$ and $Y_{tc_2} \geq \frac{1}{3}$ **then**
8: build a Type A facility for commodity t at r_0
9: **else**
10: Let r_i be the closest node to r_0 in the path $(r_0, r_1, \ldots, r_i, \ldots, r_f)$ such that we've built a Type B facility for commodity t at r_i. If we have not built any Type B facilities, let $i = f + 1$.
11: **if** $\sum_{x=0}^{i-1} \sum_{\text{children } r \text{ of } r_x \text{ such that } Y_{tr} < \frac{1}{3}} Y_{tr} \geq \frac{1}{3}$ **then**
12: build a Type B facility for commodity t at r_0
13: **else if** there is no child c of r_0 such that $Y_{tc} \geq \frac{2}{3}$ **then**
14: build a Type C facility for commodity t at r_0
15: For each client j, let S_j = the set of all edges e such that $z_{je} \geq \frac{1}{3}$.

THEOREM 3.2. *Algorithm 1 gives a 9-approximation to* MGSFL *on a tree, when for any $t, v, v', c(t, v) = c(t, v')$.*

Proof. Lemmas 3.1 and 3.2 together prove Theorem 3.2.

LEMMA 3.1. *After an execution of Algorithm 1, each client j meets each commodity $t \in I(j)$ at least once.*

Proof. Consider client j interested in commodity t. Consider some subtree (say it is rooted at r_0) such that $w_{jtr_0} \geq \frac{1}{3}$ in the LP solution, and such that for any child r of r_0, $w_{jtr} < \frac{1}{3}$. Say $(r_0, r_1, \ldots, r_f)$ is the path from r_0 to the root of the tree. Then any node in $(r_0, \ldots, r_f)$ satisfies the condition on line 5 by definition of r_0. We will show that j must meet t at some r_i.

CLAIM 3.1. *If we built a facility of Type C for commodity t at any node r_i in $(r_0, \ldots, r_f)$, then our tree for client j must include r_i.*

Proof. Suppose there is a Type C facility at node $r_i (0 < i \leq f)$. Since the condition on line 13 was true (we built a Type C facility), $Y_{tr_{i-1}} < \frac{2}{3}$. Therefore, j meets t less than $\frac{2}{3}$ in the subtree rooted at r_{i-1}. If j is a descendant of r_{i-1}, the LP solution for j must include edge (r_{i-1}, r_i) with weight at least $\frac{1}{3}$ in order for j to meet t at least once in the tree. Therefore, our solution tree for client j must include r_i. If j is not a descendant of r_{i-1}, the LP solution for j must include edge (r_i, r_{i-1}) at least $\frac{1}{3}$ in the LP solution to get to r_0 (by definition of r_0), so our

solution tree for client j will still include r_i. Therefore, j meets t at r_i.

CLAIM 3.2. *Suppose there is a facility for commodity t at node $r_i (0 < i \leq f)$, and there are no facilities for t built at any $r_g, 0 \leq g < i$. Then, our tree for client j must include r_i.*

Proof. Since we didn't build a facility at any $r_g, 0 \leq g < i$, we could not have built a Type B facility at any of these nodes. Therefore, since we didn't build a Type B facility specifically at r_0, then by line 11, $\sum_{x=0}^{i-1} \sum_{\text{children } r \text{ of } r_x : Y_{tr} < \frac{1}{3}} Y_{tr} < \frac{1}{3}$. Furthermore, since we didn't build a Type A facility at any $r_g, 0 < g < i$, the condition on line 7 must also be false for these nodes, so $Y_{tc} < \frac{1}{3}$ for all children c of r_g, $c \neq r_{g-1}$ (by definition of r_0, we know that all nodes c on the path $r_0, r_1, \ldots, r_f$ have $Y_{tc} \geq \frac{1}{3}$). We also didn't build a Type A facility at r_0, so at most one child c of r_0 has $Y_{tc} \geq \frac{1}{3}$. We didn't build a Type B facility at r_0, so there must be one child, say r_{-1} of r_0 with $Y_{tr_{-1}} \geq \frac{1}{3}$. By definition of r_0, j meets $t < \frac{1}{3}$ under each child of r_0, including r_{-1}. This gives: $\sum_{\text{children } r \neq r_{-1} \text{ of } r_0} Y_{tr} + \sum_{x=1}^{i-1} \sum_{\text{children } r \text{ of } r_x, r \neq r_{x-1}} Y_{tr} < \frac{1}{3}$, so j meets t less than $\frac{1}{3}$ under any children of this path other than r_{-1}, and j meets t less than $\frac{1}{3}$ under r_{-1}. So j meets t less than $\frac{2}{3}$ under r_{i-1}.

If j is a descendant of r_{i-1}, the tree for j in the LP solution must include edge (r_{i-1}, r_i) at least $\frac{1}{3}$, so our tree for j will include r_i. If j is not a descendant of r_{i-1}, it must pass through r_i at least $\frac{1}{3}$ to get to r_0 (by definition of r_0). So j meets t at r_i.

If there is any facility for commodity t built at r_0, j will meet t there, since j passes through r_0 by definition of r_0. Therefore, if there is a facility for commodity t built at any node in $(r_0, \ldots r_f)$, j will meet t.

If no facility is built at any node in $(r_0, \ldots r_f)$, then t must be built $\geq \frac{1}{3}$ in at most one child of r_0 (but meets j there less than $\frac{1}{3}$), and other than this subtree under r_0, t is built a total of $< \frac{1}{3}$ under r_f. So j meets $t < \frac{2}{3}$ under r_f, a contradiction since r_f is the root of the tree.

Thus there must be some facility for t built at a node in $(r_0, \ldots r_f)$ which is part of our tree for j.

LEMMA 3.2. *The cost from Algorithm 1 is at most* 9 *times the cost of the solution to the LP.*

Proof. The client costs incurred by Algorithm 1 are at most 3 times the LP client cost, since the client values z_{je} are either rounded down to 0 or multiplied by 3.

We will analyze the building costs for a single commodity, t, according to the type of facilities built.

Type A facilities: If X is the number of lowest level subtrees T_u in which the LP solution gives $Y_{tu} \geq \frac{1}{3}$ ("lowest level" means that although the LP builds t at least $\frac{1}{3}$ in the subtree T_u, the LP builds t less than $\frac{1}{3}$ under each subtree rooted at a child of u), we may build at each intersection of two such subtrees. The maximum number of such intersections equals the maximum number of non-leaf nodes with at least 2 children in a tree with X leaves. Therefore, the maximum number of such intersections is less than X. This gives us a cost less than 3 times the LP cost for building facilities.

Type B facilities: These are paid for by the "gathered" costs from the subtrees in which the commodity is built $< \frac{1}{3}$ by the LP. In other words, we are consolidating the cost of tiny fractions of built facilities over a number of subtrees and using each of these tiny fractions only towards a single facility. Since each fractional building in the LP is used to build at most one Type B facility, and since we must have at least $\frac{1}{3}$ of a building for each type B facility we build, this cost is at most 3 times the LP cost for building facilities.

Type C facilities: These are paid for by the $> \frac{1}{3}$ building below the node at which the Type C facility is built. Again, we pay at most 3 times the LP cost for building facilities.

We now bound the total cost. The fractional building cost at a node u in the LP may be used to build a Type C facility only at ancestors of u. No other facility will be built at any node whose ancestor contains a Type C facility (according to the condition at line 6 of the algorithm). Therefore, if the same cost at u is also used towards a Type B facility, the Type B facility must be located at an ancestor of the Type C facility. By the condition at line 5, facilities of any type are only built at the root of a subtree in which there is $> \frac{1}{3}$ built by the LP, so the LP must build the commodity $> \frac{1}{3}$ in the subtree rooted at the Type C facility. However, the cost of a Type B facility comes entirely from subtrees with $< \frac{1}{3}$ built by the LP. Therefore, the fractional building at u cannot be used towards both a Type C and a Type B facility. This means that our building costs are at most the LP building costs times [the cost for building Type A facilities times the maximum of (Type B building costs, Type C building costs)] ≤ 9 times the LP building costs.

We can use the probabilistic embedding technique from [7] to convert the problem on general metrics to the problem on trees, adding only a factor $O(\log n)$ to the approximation ratio. Therefore, this algorithm gives an $O(\log n)$-approximation on general metrics.

4 General MGSFL

In this section, we give a $2^{O(\sqrt{\log n}\log\log n)}$ approximation algorithm for the general MGSFL problem. Our algorithm uses a generalization of the elegant randomized rounding algorithm of [8] for Group Steiner Tree that may be of independent interest.

Height reduction. First, we will convert the input tree into a tree with at most $\sqrt{\log n}$ levels. This can be done as in Corollary 1.1, giving up a factor of $2^{\sqrt{\log n}}$ in the approximation ratio.

Solving LP and preprocessing. Next, we solve the LP applied to the tree of reduced height. Before rounding, we ensure that each of our variables has a value $\geq \frac{1}{2n}$, and each x_{jtu} variable has a value $< \frac{1}{8}$. To make sure all values are at least $\frac{1}{2n}$, we will (a) set any value below $\frac{1}{2n}$ to 0, and (b) double all remaining values. All LP constraints still hold after each of these steps except for Constraint 2.2, which will break after step (a). However, we've only removed at most $n \cdot \frac{1}{2n} = \frac{1}{2}$ from the left hand side, so by doubling the values in step (b), we get back to a valid (although non-optimal) solution. We have at most doubled the value of the objective function, and we are now assured that each value is at least $\frac{1}{2n}$. To ensure that each $x_{jtu} < \frac{1}{8}$, we proceed as follows. For any j, t such that there exists some $x_{jtu} \geq \frac{1}{8}$, pick one such u. Set $x_{jtv} = 0$ for all v, set $y_{tu} = 1$, and set $x_{je} = 1$ for all $e \in P_{ju}$. Now we have guaranteed that j and t will meet at node u, and we have at most multiplied the objective function by an additional 8.

Rounding. The core of the algorithm is the following randomized rounding procedure. We set new variables w'_{jte}, x'_{jtu}, y'_{tu}, and z'_{je} (each will be 0 or 1). Initially, set $w'_{jte} = 0$. Follow the steps below.

1. For each edge e, assign a random value $r_e \in [0, 1]$.
2. For each client j, commodity $t \in I(j)$, and edge e . Assume the path from the root to e is $e_1, e_2, \ldots, e_p = e$. Set $w'_{jte} = 1$ if and only if $w_{jte_1} \geq r_{e_1}$ and $\frac{w_{jte_\alpha}}{w_{jte_{\alpha-1}}} \geq r_{e_\alpha}$ for all $\alpha \in 2, \ldots p$.
3. For each client j, commodity $t \in I(j)$, leaf u: set $x'_{jtu} = 1$ if and only if $w'_{jte} = 1$ for e adjacent to leaf u.

Finally, set $y'_{tu} = 1$ if and only if $y_{tu} = 1$ or there is some j such that $x'_{jtu} = 1$. Set $z'_{je} = 1$ if and only if $z_{je} = 1$ or e is on the path from j to some u with $x'_{jtu} = 1$.

The above algorithm does not guarantee that each client meets all commodities in its interest set (or even that all commodities have been built). Therefore, we will repeat the algorithm, including the union of the results, until all requirements are met.

Our analysis proceeds as follows. We will start in Lemma 4.1 by analyzing the expected cost of our solution. Then, in Lemma 4.2, we show that there is a high probability that a given client will meet a given commodity in its interest set. Finally, we compute how many times we can expect to repeat the algorithm to achieve an arbitrarily high probability that each client meets all commodities in its interest set. Our total expected cost is the expected cost of a single iteration times the number of iterations.

LEMMA 4.1. *From the above rounding steps 1 to 3, we get*

$$E\left[\sum_{j,e} z'_{je} \cdot d(e) + \sum_{t,u} y'_{tu} \cdot c(u,t)\right] \leq 2^{O(\sqrt{\log n}\log\log n)} \cdot \sum_{j,e} z_{je} \cdot d(e) + \sum_{t,u} y_{tu} \cdot c(u,t)$$

Proof. We start by analyzing this for a single y'_{tu}. Suppose the path to u is made up of edges $e_1, \ldots, e_p$. For ease of notation, let $v_{j1} = w_{jte_1}$, and $v_{j\alpha} = \frac{w_{jte_\alpha}}{w_{jte_{\alpha-1}}}$ (for each client j interested in commodity t). With this modified notation, $y'_{tu} = 1$ if and only if there exists some j such that $v_{j\alpha} \geq r_{e_\alpha}$ for all α. Furthermore, $\prod_{\alpha=1}^{p} v_{j\alpha} \leq y_{tu}$, since w_{jte_p} is the only term that does not cancel (which is exactly x_{jtu}), and by LP Constraint 2.3, $x_{jtu} \leq y_{tu}$. We will artificially increase the final value (v_{jp}) for each client so that each client has exactly $\prod_{\alpha=1}^{p} v_{j\alpha} = y_{tu}$. This will only increase the expected value of y'_{tu}.

Now we are looking at a series of p-dimensional points $\{V_j = (v_{j1}, \ldots, v_{jp})\}$ that each satisfy $\prod_{\alpha=1}^{p} v_{j\alpha} = y_{tu}$, plus one randomly chosen p-dimensional point $R = (r_{e_1}, \ldots r_{e_p})$. $y'_{tu} = 1$ if and only if there exists some j just that $R \leq V_j$ in each dimension. We can upper bound this probability by the probability that R will lie underneath the p-dimensional curve $\prod_{\alpha=1}^{p} x_\alpha = y_{tu}$ where each variable ranges from y_{tu} to 1. This is illustrated for a 2-hop path in Figure 1.

The area above the curve is given by the following

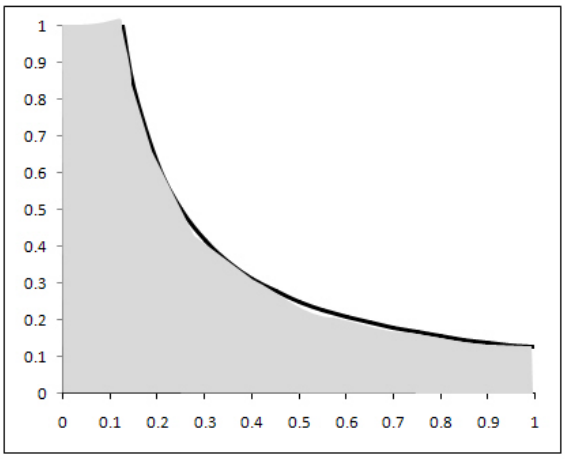

Figure 1: Analyzing the probability that a commodity y will be built at some leaf of a tree with depth 2. For the data plotted in this graph, $y_{tu} = \frac{1}{8}$. Each client meets t with probability y_{tu} in the LP solution. Any possible values of v_{j1} and v_{j2} for clients j meetings commodity t at this node are represented by points along this curve. t will be built here only if our random point $R = (R_1, R_2)$ lies in the gray area under this curve.

integration.

$$\begin{aligned}
&\int_{y_{tu}}^{1} \int_{\frac{y_{tu}}{x_p}}^{1} \int_{\frac{y_{tu}}{x_p x_{p-1}}}^{1} \cdots \\
&\qquad \int_{\frac{y_{tu}}{x_p x_{p-1} \cdots x_3}}^{1} \int_{\frac{y_{tu}}{x_p x_{p-1} \cdots x_3 x_2}}^{1} dx_1 dx_2 \ldots dx_{p-1} dx_p \\
&= \quad 1 - y_{tu} - \sum_{i=1}^{p} (-1)^i \left(\frac{y_{tu}}{i!}\right) \ln^i \left(\frac{1}{y_{tu}}\right) \\
&= \quad 1 - y_{tu} - \sum_{i=1}^{p} \left(\frac{y_{tu}}{i!}\right) \ln^i \left(\frac{1}{y_{tu}}\right)
\end{aligned}$$

Subtracting the above expression from 1 will give the area under the curve, so our actual upper bound on the expectation of building commodity t at node u is

$$\begin{aligned}
& y_{tu} + \sum_{i=1}^{p} \left(\frac{y_{tu}}{i!}\right) \ln^i \left(\frac{1}{y_{tu}}\right) \\
= \quad & y_{tu} \cdot \left(1 + \sum_{i=1}^{p} \frac{\ln^i \left(\frac{1}{y_{tu}}\right)}{i!}\right) \\
\leq \quad & y_{tu} \cdot \left(1 + \sum_{i=1}^{p} \frac{\ln^i \left(\frac{1}{y_{tu}}\right)}{i!}\right)
\end{aligned}$$

After solving the LP, we took steps to ensure that all y_{tu} values were at least $\frac{1}{2n}$, so $\ln^i(1/y_{tu})/i!$ is at most $\ln^i(2n)/i!$, which by Stirling's approximation is at most $((e \ln n)/i)^i$. Since $1 \leq i \leq p$, and p is the height of the tree which is at most $O(\sqrt{\log n})$, $\ln^i(1/y_{tu})/i!$ is maximized at $i = p$ yielding an upper bound of $(e\sqrt{\log n})^{\sqrt{\log n}}$. We thus have the following upper bound on the expected cost of building commodity t at node u.

$$\begin{aligned}
E[y'_{tu}] \quad &\leq \quad y_{tu} \cdot \left(1 + \sum_{i=1}^{p} \frac{\ln^i \left(\frac{1}{y_{tu}}\right)}{i!}\right) \\
&\leq \quad (p+1) \left(e\sqrt{\log n}\right)^{\sqrt{\log n}} \cdot y_{tu} \\
&= \quad 2^{O(\sqrt{\log n} \log\log n)} \cdot y_{tu}
\end{aligned}$$

We next bound $E[z'_{je}]$ in terms of z_{je}. First, assume e is an edge adjacent to a leaf u, and assume the part of the path from j to u is made up of edges $e_1, \ldots, e_p$. As above, let $v_{t1} = w_{jte_1}$ and $v_{t\alpha} = \frac{w_{jte_\alpha}}{w_{jte_{\alpha-1}}}$ (for each commodity t in which client j is interested). Then, $z'_{je} = 1$ if and only if there exists some t such that $v_{t\alpha} \geq r_{e_\alpha}$ for all α. $\prod_{\alpha=1}^{p} v_{t\alpha} \leq z_{je}$ (as above), since w_{jte_p} is the only term that is not cancelled (which is exactly x_{jtu}), and by LP Constraint 2.1, $x_{jtu} \leq z_{je}$. We will artificially increase the final value (v_{tp}) for each commodity so that each commodity has exactly $\prod_{\alpha=1}^{p} v_{t\alpha} = z_{je}$. Now we can solve the same integral as above to get $E[z'_{je}] = 2^{O(\sqrt{\log n} \cdot \log\log n)} \cdot z_{je}$.

Now we need to consider an edge e that is not adjacent to a leaf, and may therefore be on the path to more than one leaf. Assume the edges on the path from j to e are $e_1, \ldots, e_p = e$. We set $z'_{je} = 1$ if and only if e is on the path to some u with $x'_{jtu} = 1$. In order for any one of these x'_{jtu} to be 1, it must be the case that $w_{jte_1} \geq r_{e_1}$, and $\frac{w_{jte_\alpha}}{w_{jte_{\alpha-1}}} \geq r_{e_\alpha}$ for all $\alpha = 2 \ldots p$. For any commodity t, the product of these values is exactly $w_{jte_p} = w_{jte} \leq z_{je}$ (by definition of w). Now we can apply the same logic as before to show $E[z'_{je}] = 2^{O(\sqrt{\log n} \cdot \log\log n)} \cdot z_{je}$

Adding these together and applying linearity of expectation, we complete the proof of the theorem.

$$\begin{aligned}
& E\left[\sum_{j,e} z'_{je} \cdot d(e) + \sum_{t,u} y'_{tu} \cdot c(u,t)\right] \\
& \leq 2^{O(\sqrt{\log n} \log\log n)} \cdot \sum_{j,e} z_{je} \cdot \left(d(e) + \sum_{t,u} y_{tu} \cdot c(u,t)\right)
\end{aligned}$$

Next, we will use Janson's inequality to bound the probability that a given client-commodity pair does not

meet in the rounded solution. This probability is low enough that we can bound the number of expected iterations before all requirements are met. We start by defining Janson's inequality. We borrow the following notation from [8]. Let Ω be a universal set, and $R \subseteq \Omega$ determined by the experiment in which each element $r \in \Omega$ is independently included in R with probability p_r. Let A_i be subsets of Ω, and denote by B_i the event that $A_i \subseteq R$. Write $i \sim j$ if B_i and B_j are not independent. Define $\Delta = \sum_{i \sim j} \Pr[B_i \cap B_j]$ (the sum is over ordered pairs). Let $\mu = \sum_i \Pr[B_i]$, and ϵ be such that $\Pr[B_i] \leq \epsilon$ for all i.

THEOREM 4.1. (JANSON'S INEQUALITY) *With the notation as above, if $\Delta \geq \mu(1-\epsilon)$, then $\Pr[\cap_i \overline{B_i}]$ is at most $e^{-\mu^2(1-\epsilon)/(2\Delta)}$.*

The proof of the next lemma is adapted from [8].

LEMMA 4.2. *For a given client j and commodity $t \in I(j)$, assuming $x_{jtu} < \frac{1}{8}$ for all u, the above steps 1 to 3 will set x' values that satisfy LP Constraint 2.2 ($\sum_u x_{jtu} \geq 1$) with probability at least $1 - e^{\frac{-7}{16\sqrt{\log n}}}$.*

Proof. We will set the variables defined above for use with Janson's inequality. First find the first edge f along the path from j to the root with $z_{jf} = 0$. Ω will be the set of all edges (not on the path from j to f) in the subtree rooted at f. For each edge $e \in \Omega$ we will define p_e as follows. Assume e has parent edge e' (e' is the next edge along the path from e to f). If e is adjacent to a_{ue}, $p_e = w_{jte}$. Else, $p_e = \frac{w_{jte}}{w_{jte'}}$.

Define a subset A_u for each node u that has $x_{jtu} > 0$: A_u is the set of all edges along the path from a_{ju} to u. By definition of A_u and w, $w_{jte} > 0$ for all $e \in A_u$. Then B_u (the event that $A_u \subseteq R$) is the event that for node u (with $x_{jtu} > 0$) we included all of the edges from a_{ju} to u.

Next, in Claim 4.1, we will show that the bound we'll get from Janson's inequality will be useful. Specifically, Janson's inequality will bound $\Pr\left[\bigcap_i \overline{B_i}\right]$, and we want to bound the probability that a given client-commodity pair will fail to meet, or $\Pr\left[\bigcap_u (x'_{jtu} = 0)\right]$.

CLAIM 4.1. *The probability of event B_u is exactly the probability that $x'_{jtu} = 1$. This immediately implies that $\Pr\left[\bigcap_u \overline{B_u}\right] = \Pr\left[\bigcap_u (x'_{jtu} = 0)\right]$.*

Proof. First, if there is some edge e in B_u with $w_{jte} = 0$, then (by definition of w) there is no u in the subtree rooted at e with $x_{jtu} > 0$. This contradicts our selection of B_u. Therefore, we can assume that $w_{jte} > 0$ for all $e \in B_u$.

Next, recall that $x'_{jtu} = 1$ if an only if $w_{jte_1} \geq r_{e_1}$ and $\frac{w_{jte_\alpha}}{w_{jte_{\alpha-1}}} \geq r_{e_\alpha}$ for all $\alpha \in 2, \ldots p$ (where $e_1 \ldots e_p$ are the edges in B_u, sorted by the distance from the root). Therefore, $\Pr[x'_{jtu} = 1] = w_{jte_q} \cdot \prod_{\alpha=2}^{p} \frac{w_{jte_\alpha}}{w_{jte_{\alpha-1}}}$. Similarly, the probability that each edge is included in B_u is the product of the p_e values, which is also $w_{jte_q} \cdot \prod_{\alpha=2}^{p} \frac{w_{jte_\alpha}}{w_{jte_{\alpha-1}}}$.

In order to apply Janson's inequality, we need to show $\Delta \geq \mu(1-\epsilon)$. Therefore, we will bound Δ and μ in the next two claims. Recall, $\Delta = \sum_{u \sim v} \Pr[B_u \cap B_v]$, and $\mu = \sum_u \Pr[B_u]$.

CLAIM 4.2. $\Delta = \Theta(\sqrt{\log n})$

Proof. For this proof, let T_e be the subtree rooted at edge e. Recall that P_{uv} is the path from u to v, and a_{uv} is the least common ancestor of u and v.

For each ordered pair $u \sim v$, if edge e_u is the edge adjacent to u and e_v is the edge adjacent to v, we have $\Pr[B_u \cap B_v] = \frac{w_{jte_u} w_{jte_v}}{w_{jte}}$ where e is the root-side edge adjacent to a_{uv}. We know that such an edge exists because $u \sim v$. By definition of w and the fact that u and v are leaves, this gives $\Pr[B_u \cap B_v] = \frac{x_{jtu} x_{jtv}}{w_{jte}}$. We can write $\Delta = \sum_u \sum_v \frac{x_{jtu} x_{jtv}}{w_{jte}}$. But for any e, $\sum_{(v \in T_e)} x_{jtv} = w_{jte}$ (by definition of w). This gives (since the tree has depth at most $O(\sqrt{\log n})$)

$$\begin{aligned}
\Delta &\leq \sum_u x_{jtu} \sum_{(e \in P_{ua_{ju}})} \frac{1}{w_{jte}} \sum_{(v \in T_e)} x_{jtv} \\
&= \sum_u x_{jtu} \sum_{(e \in P_{ua_{ju}})} 1 \\
&\leq \sum_u x_{jtu} O(\sqrt{\log n}) \\
&= O(\sqrt{\log n})
\end{aligned}$$

CLAIM 4.3. $\mu = 1$

Proof. For a given B_u with edges $e_1, \ldots e_p$ on the path from a_{ju} to u,

$$\Pr[B_u] = w_{jte_1} \cdot \prod_{\alpha=2}^{p} \frac{w_{jte_\alpha}}{w_{jte_{\alpha-1}}} = w_{jte_p} = x_{jtu}$$

By LP Constraint 2.2, $\sum_u x_{jtu} = 1$. Therefore, $\mu = 1$.

Now, if we set $\epsilon = \frac{1}{8}$, then $\Pr[B_u] \leq \epsilon$ for all u by the stipulations of Lemma 4.2. We also know $\Delta \geq \frac{7}{8} = \mu(1-\epsilon)$ for large enough n, so Janson's inequality gives us:

$$\Pr\left[\bigcap_u \overline{B_u}\right] \leq e^{\frac{-\mu^2(1-\epsilon)}{2\Delta}}$$

Or in terms of our problem:

$$\Pr\left[\bigcap_u (x'_{jtu} = 0)\right] \leq e^{\frac{-7}{16\sqrt{\log n}}}$$

Thus, given some pair j, t, the probability that they don't meet is at most $e^{\frac{-7}{16\sqrt{\log n}}}$. If we repeat the algorithm X times, the probability that the pair won't meet in any iteration is at most $\left(e^{\frac{-7}{16\sqrt{\log n}}}\right)^X$. Therefore, in order ensure that after X iterations the probability that *any* pair fails to meet is less than ϵ, we set X so that

$$\epsilon > nm\left(e^{\frac{-7X}{16\sqrt{\log n}}}\right) \geq \sum_j \sum_{t \in I(j)} e^{\frac{-7X}{16\sqrt{\log n}}}$$

$$X \geq \frac{16\sqrt{\log n}}{7 \log e}\left(\log(nm) - \log \epsilon\right)$$

Thus, we can set X, the number of iterations, to anything larger than

$$\begin{aligned}
&\frac{16\sqrt{\log n}}{7 \log e}\left(\log(nm) - \log \epsilon\right) \\
&< 3\sqrt{\log n}\log(nm) + 2\sqrt{\log n}\log\left(\frac{1}{\epsilon}\right) \\
&= O(\sqrt{\log n}\left(\log(nm) + \log\left(\frac{1}{\epsilon}\right)\right)
\end{aligned}$$

Our number of iterations is thus poly-logarithmic in the size of the input (and in ϵ), and our approximation ratio is computed as follows.

- We lost a factor $2^{O(\sqrt{\log n})}$ by converting the tree to one with at most $O(\sqrt{\log n})$ levels.
- We lost a factor of 16 by using only LP values that were between $\frac{1}{2n}$ and $\frac{1}{8}$.
- Our expected cost was at most $2^{O(\sqrt{\log n}\log\log n)}$ times the cost of the optimal LP solution on the converted tree (ignoring the very large and very small values from the LP solution).

This gives a final approximation ratio $2^{O(\sqrt{\log n}\log\log n)}$ on trees. We can apply the results from [7] to convert any metric into a tree metric with distortion $O(\log n)$, so this also gives an approximation ratio $2^{O(\sqrt{\log n}\log(\sqrt{\log n}))}$ on general metrics.

As noted earlier, our goal for MGSFL was an algorithm with a poly-logarithmic approximation ratio, since this would be close the known $\Omega(\log^{2-\epsilon} m)$ hardness result from [11]. $2^{O(\sqrt{\log n}\log\log n)}$ does not achieve this goal, but it is possible that a more refined algorithm could obtain a tighter bound. Our algorithm and analysis does not take all properties of the LP solution into account. For instance, although a commodity may have significantly higher expected building cost for a specific client at a single node then the LP cost, this would only occur if the LP built the same commodity some amount at each of many other nearby nodes. We may be able to account for the higher expected building cost by carefully considering *why* this commodity was built at a variety of nearby locations.

5 Laminar Client Interest Sets

In this section, we examine another special case of MGSFL. Instead of restricting the building cost or distance functions, we consider the general problem with restrictions on the interest sets. Specifically, we require that the interest sets form a *laminar* collection; that is, given any two clients j and j' with $|I(j)| \leq |I(j')|$, either $I(j) \subseteq I(j')$ or else $I(j) \cap I(j') = \emptyset$. Since the hardness result for Group Steiner Tree applies to MGSFL even with a single client, MGSFL with laminar interest sets is also $\Omega(\log^{2-\epsilon} m)$-hard to approximate.

For MGSFL with laminar interest sets, our approximation algorithm consists of these high-level steps.

1. Convert the original tree into a tree of depth at most $O(\log n)$.
2. Solve the LP on the new tree.
3. Use the LP solution to convert the problem into a series of revised problems in which all clients are located at the root of the input tree.
4. Solve each revised problem.

Step 1 can be done using Theorem 1.1, giving up only a constant factor in the approximation ratio. Step 2 will then give us a fractional solution which has cost at most a constant times the cost of the original optimal solution. In Lemma 5.1, we show that, at the expense of a factor proportional to the height of the tree, we can reduce to the case where all the clients are located at the root. Step 3 will thus cost us an extra factor $O(\log n)$ in the approximation ratio. In the remainder of this section, we focus on step 4 of the algorithm, which will cost an extra factor $O(\log^2 n \log m)$ in the approximation ratio, as shown in Theorem 5.1. This yields an $O(\log^3 n \log m)$-approximation for MGSFL with laminar interest sets on tree metrics.

LEMMA 5.1. *If we can find a solution to* MGSFL *when all clients are located at the root which costs at most R times the optimal cost of a fractional solution to the same problem, then we can find an $O(h(T) \cdot R)$*

approximation to MGSFL*, where $h(T)$ is the height of the tree, in which each client may be located at any leaf.*

Proof. Our algorithm proceeds as follows.

1. Solve the LP.
2. For each client j, consider the set of edges e on the path from the root to the location of j. Find the closest such edge e to the root such that $x_{je} \geq \frac{1}{2}$. Set the location of j to the root endpoint of e.
3. For each internal node v, find an R-approximation for MGSFL for the subtree rooted at v (call this T_v), for only the clients now located at v, and only the commodities in the interest sets of clients now located at v.

Consider some level l of the tree, and the set of subtrees $T_{l1}, T_{l2}, \ldots T_{ld}$ rooted at the nodes at level l. Now, consider the same linear program with the following additional constraints for any client j which has been restricted to subtree T_{li}

$$\begin{aligned} x_{jtu} &= 0 \qquad \text{for all } u \text{ not in } T_{li}, \text{ for all } t \in I(j) \\ z_{je} &= 0 \qquad \text{for all } e \text{ not in } T_{li} \end{aligned}$$

Now, if we take the solution to the original LP, multiply all values by 2, and set the appropriate x_{jtu} and z_{je} values to 0 as required by the extra constraints, we claim that (a) we have a valid solution to the LP with the extra constraints, and (b) the cost of our new solution is at most twice the cost of the original solution.

We note that (b) is true because

$$\begin{aligned} & \sum_{j,e} 2 \cdot z_{je} \cdot d(e) + \sum_{u,t} 2 \cdot y_{tu} \cdot c(u,t) \\ = \quad & 2 \cdot \left(\sum_{j,e} z_{je} \cdot d(e) + \sum_{u,t} y_{tu} \cdot c(u,t) \right) \end{aligned}$$

For (a), we will check each constraint. The extra constraints are obeyed by construction, and the non-negativity constraints from the original LP clearly still hold, since we are not increasing any values that were previously set to 0, and we are not decreasing any values beyond 0. For Constraint 2.1, if $e \in T_{li}$, then we have increased x_{je} by a factor 2, and we have at most increased each x_{jtu} by a factor of 2, so the constraint still holds. If $e \notin T_{li}$, then we have set $z_{je} = 0$. However, if $e \in P_{ju}$, and $e \notin T_{li}$, then $u \notin T_{li}$, so all applicable x_{jtu} have also been set to 0. Next we'll examine Constraint 2.2. In the original LP solution, $x_{je} < \frac{1}{2}$ for the edge e that crosses out of s_{li} towards the root. Therefore, by Constraint 2.1, $\sum_{u \notin T_{li}} x_{jtu} < \frac{1}{2}$, so $\sum_{u \in T_{li}} x_{jtu} \geq 1 - \frac{1}{2} = \frac{1}{2}$ (by Constraint 2.2). We have multiplied all x_{jtu} for $u \in T_{li}$ by 2, so we now have $\sum_{u \in T_{li}} x_{jtu} \geq 1$. Finally, check Constraint 2.3. We have not dropped any y_{tu} down to 0, and we multiplied all values by 2, so this constraint still holds.

If we were to solve this modified LP restricted to only the clients now located at roots of the subtrees $s_{l1}, \ldots s_{ld}$, this would be a (fractional) solution to our revised MGSFL instance for level l, and would cost at most 2 times the optimal cost of the original instance.

Summing over all levels gives:

$$\begin{aligned} & \sum_{v} \text{cost(optimal fractional solution for revised problem on } T_v) \\ = \quad & \sum_{l=1}^{h(T)} \text{cost(optimal fractional solution for revised problem at level } l) \\ \leq \quad & \sum_{l=1}^{h(T)} 2 \cdot \text{cost(optimal fractional solution for original problem)} \\ = \quad & 2 \cdot h(T) \cdot \text{cost(optimal fractional solution for original problem)} \\ \leq \quad & 2 \cdot h(T) \cdot \text{cost(optimal solution for original problem)} \end{aligned}$$

Since we are assuming we can find a solution to the revised problem that costs at most R times the cost of the optimal fractional solution to the revised problem, this gives

$$\begin{aligned} & \sum_{v} \text{cost(our solution for the revised problem on } T_v) \\ \leq \quad & R \cdot \sum_{v} \text{cost(optimal fractional solution for revised problem on } T_v) \\ \leq \quad & 2 \cdot h(T) \cdot \text{cost(optimal solution for original problem)} \end{aligned}$$

THEOREM 5.1. *Given an instance of* MGSFL *on a tree in which (a) all clients are located at the root, and (b) the interest sets are laminar, we can achieve an $O(\log^2 n \log m)$ approximation in polynomial time.*

Proof. In this proof, we refer to the *laminar forest* L of clients. By our restriction on the interest sets, any two clients j and j' with $|I(j)| \leq |I(j')|$ have $I(j) \subseteq I(j')$ or $I(j) \cap I(j') = \emptyset$. First, define an arbitrary ordering of the clients, $j \prec j'$. Now, we place the clients as nodes in the laminar forest such that j is a descendant of j' in

the forest if and only if $|I(j) \subset I(j')|$ OR $[I(j) = I(j')$ AND $j \prec j']$. Given the order $\prec$, this forest is unique.

We will assume the following lemma, to be proved later.

LEMMA 5.2. *We can decompose the laminar forest L of clients into groups $G_1, G_2, \ldots, G_a$, $S_1, S_2, \ldots S_a$ (for some a) that obey the following rules:*

1. *For each commodity t, at most $O(\log n)$ clients in $\bigcup_i G_i$ are interested in t.*
2. *There is some mapping π from all clients in S_i to clients in $G_{i-1} \cup S_{i-1}$ such that (a) $\pi(u)$ is an ancestor of u, and (b) $|\{v : \pi(v) = u\}| \leq 2$ for each u.*
3. *There is some mapping φ from each client in S_i to some client in G_{i-1} such that $\varphi(u)$ is a descendant of $\pi(u)$ and an ancestor of u.*

Assuming Lemma 5.2, we use the Group Steiner Tree algorithm of [8] to independently solve for each node in $\bigcup_i G_i$. The Group Steiner Tree problem does not include building costs. In order to account for the fact that our building costs depend on the location as well as on the commodity, we solve the problem on a slightly adjusted tree. For each t, v for which $c(t, v) < \infty$, we add a leaf from v with length $c(t, v)$, and we allow t at the new leaf instead of at v. Since we are solving for a single client, the additional cost to include the new edge is exactly the same as the cost of building the facility. The algorithm of [8] achieves an $R = \log n \log m$ approximation with respect to the optimal fractional solution. We now prove that our algorithm yields an $O(R \log n) = O(\log^2 n \log m)$-approximation to the MGSFL instance.

By Requirement 1 for the groups G_i, we build each commodity at most $O(\log n)$ times. Since any solution (including any optimum fractional solution) must build each commodity at least once, and since each Group Steiner Tree iteration finds an R approximation including the building cost edges, this gives a building cost $\leq O(R \log n)$ times the optimal building cost.

Each client in $\bigcup_i G_i$ is paying at most R times its optimal fractional cost if it were the only client, since MGSFL with a single client is exactly equivalent to Group Steiner Tree. It cannot do better than this when adjusting for other clients, so the total client cost for $\bigcup_i G_i$ is at most the optimal fractional client cost.

Now consider the cost for a client $u \in S_i$ for some i. Since $\varphi(u)$ is an ancestor of u in L, u is interested in a subset of the commodities in which $\varphi(u)$ is interested. Therefore, we can handle u by sending it to the same places as $\varphi(u)$ and pay at most the same cost we spent for $\varphi(u)$. Since $\varphi(u) \in G_{i-1}$, the cost we spend for u is at most R times the optimal fractional cost for $\varphi(u)$. Since $\varphi(u)$ is a descendant of $\pi(u)$ in L, $\pi(u)$ needs to access a superset of the commodities required for $\varphi(u)$, so the optimal fractional cost for $\pi(u)$ is at least the optimal fractional cost for $\varphi(u)$. This gives us the following cost analysis for clients in $\bigcup_i S_i$. "opt" refers to the optimal fractional cost, "opt(u)" refers to the optimal fractional cost for Group Steiner Tree with client u, and "cost" refers to our cost.

$$\begin{aligned}
\text{cost}(\bigcup_i S_i) &= \sum_i \sum_{u \in S_i} \text{cost}(u) \\
&\leq \sum_i \sum_{u \in S_i} \text{cost}(\varphi(u)) \\
&\leq \sum_i \sum_{u \in S_i} R \cdot \text{opt}(\varphi(u)) \\
&\leq R \cdot \sum_i \sum_{u \in S_i} \text{opt}(\pi(u)) \\
&\leq R \cdot \sum_i \sum_{v \in G_{i-1} \cup S_{i-1}} 2 \cdot \text{opt}(v) \\
&\leq 2R \cdot \sum_i \sum_{v \in G_{i-1} \cup S_{i-1}} \text{opt}(v) \\
&\leq 2R \cdot \text{opt}
\end{aligned}$$

Proof of Lemma 5.2. Start with $S_1 = \emptyset$, $G_1 =$ the roots of the forest. To create S_{i+1}, start with the leaves of $S_i \cup G_i$ and move backwards up the tree. Following this order, for each node $u \in S_i \cup G_i$, look at all of the nearest remaining descendants of u, and include in S_{i+1} the two that have the most descendants (as long as u has any remaining descendants). Once we have included two nodes (if available) for each node in $S_i \cup G_i$ to become part of S_{i+1}, let G_{i+1} be the set of all (not yet included) children of nodes in $G_i \cup S_i \cup S_{i+1}$.

An example of this partitioning procedure is illustrated in Figure 2. Assuming the nodes in the box in the figure are part of $S_i \cup G_i$, the figure shows the order in which nodes are included into S_{i+1} along with the choice of π. The function π maps each non-boxed node numbered i to the boxed node numbered i. Each node marked with an x will be included in G_{i+1}.

We now show that the above partitioning procedure, together with π and φ to be defined, obeys all of the desired requirements.

Requirement 2 is obeyed by construction and by the following definition of π. When we pick a node $u \in S_i$ explicitly as a descendant of some node $v \in S_{i-1} \cup G_{i-1}$, let $\pi(u) = v$. We pick at most two such nodes for each $v \in S_{i-1} \cup G_{i-1}$, so $|\{u : \pi(u) = v\}| \leq 2$.

Requirement 3 is obeyed by construction and by the following definition of φ. We picked the set of all

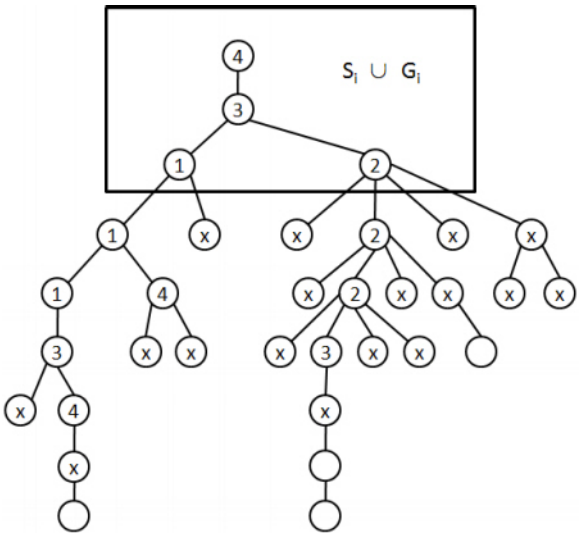

Figure 2: This figure shows how the partitioning algorithm described in the proof of Lemma 5.2 would choose S_{i+1}, G_{i+1}, and π given $S_i \cup G_i$. Assuming the nodes in the box are part of $S_i \cup G_i$, this figure shows the order in which nodes are included into S_{i+1} along with the choice of π. The function π maps each non-boxed node numbered i to the boxed node numbered i. Each node marked with an x will be included in G_{i+1}.

unpartitioned children of nodes in S_i to become the set G_i. Therefore, any descendant of a node in S_i (in particular, the nodes chosen to become part of S_{i+1}) will certainly have an ancestor in G_i. In particular, for any $u \in S_{i+1}$, consider the path from $\pi(u)$ to u. Let $\varphi(u) =$ the node along this path that is part of G_i.

To show that we obey Requirement 1, we must show that there are most $O(\log n)$ members of $\bigcup_i G_i$ along any path from a root to a leaf. Consider any path P from a root to a leaf. Each time we create a group G_i, it must contain at least one node in P, since we always add the next children to the current partitions. Each time we create a group S_i (before we've completely partitioned P), either $S_i \cap P$ must contain at least $2 \cdot |(S_i \cup G_i) \cap P|$ nodes or else some node in $(S_i \cup G_i) \cap P$ has another child (not in P) with more descendants than the next node in P. We will first analyze how often this second scenario may occur.

Suppose there are x times when a node in $(S_i \cup G_i) \cap P$ has a child not in P with more descendants than the next node in P. Let $U = u_1, \ldots u_x$ represent the nodes in $(S_i \cup G_i) \cap P$, where u_i is a descendant of u_j if $i < j$. Let $D_P(u_i)$ refer to the descendants of u_i via its child in P, and let $D_N(u_i)$ refer to the descendants of u_i via the other child (not in P) which has more descendants than the next node in P.

Since we are only counting occurrences before we've completely partitioned P, it must be true that $D_P(u_1) \geq 1$, so $D_N(u_1) \geq 1$. Since u_1 is a descendant of u_2, $D_P(u_2) \geq D_P(u_1) + D_N(u_1) + 1 \geq 3$, so $D_N(u_2) \geq 3$. Similarly, $D_N(u_i) \geq D_P(u_i) \geq 2D_P(u_{i-1}) + 1$.

We know that the total number of nodes in the graph is n, so we can solve the recurrence relation $n \geq D_P(u_i) \geq 2D_P(u_{i-1}) + 1$ to find that $x \leq \lceil \log n \rceil$. Therefore, there are at most $\lceil \log n \rceil$ times when some node in $(S_i \cup G_i) \cap P$ has another child (not in P) with more descendants than the next node in P, and the rest of the time $S_i \cap P$ must contain at least $2|(S_i \cup G_i) \cap P|$ nodes.

Now, we can separate the nodes in P into $k \leq \log n + 2$ sections, $P_1, \ldots P_k$ divided by the at most $\lceil \log n \rceil$ nodes in U described above. Let x_j denote $|(\bigcup_i G_i) \cap P_j|$ for all j. Within each of these sections, $|S_i \cap P| \geq 2 \cdot |(S_i \cup G_i) \cap P|$. Therefore, $|P_j| \geq \sum_{a=1}^{x_j} \sum_{b=1}^{a} 2^i$. After each section, we have one of the nodes in U, which has at least as many descendants of some child not in P as of some child in P. This means that there are at least j times as many nodes as the number of nodes in section P_j, since at least as many nodes exist under each node in U that is a descendant of the section P_j. This gives us the following bound for any j:

$$n \quad \geq \quad j \sum_{a=1}^{x_j} \sum_{b=1}^{a} 2^i \quad > \quad j 2^{x_j} \quad > \quad 2^{x_j}$$

Thus $\log n > x_j$. Since we already bounded the number of P_j as at most $\log n + 2$, this gives a total bound $\sum_j x_j < \log^2 n$.

We thus have an $O(\log^3 n \log m)$ approximation algorithm for MGSFL with laminar interest sets on tree metrics. As in the previous versions, we can apply the metric embeddings from [7] to get an $O(\log^4 n \log m)$ approximation bound on this version for general metrics.

6 1-Median Variant

We also consider a simple variant of MGSFL in which each commodity may only be built once. This is the equivalent of saying $c(t, v) = c(t, u)$ for all v, u, and $c(t, v) > n^2 \cdot \max_{(e \in E)} d(e)$ for all t, v, so that although we must build each commodity once in a feasible solution, it would always be better to send all interested clients to this one location than to build the commodity a second time. We show how to optimally solve the problem on a tree in which all clients are located at the leaves, which gives an $O(\log n)$

approximation for general metrics. We note that this algorithm is a generalization of an algorithm from [2] for solving the multicast push-pull data dissemination problem with aggregation.

We will require the following notation for our algorithm definition.

- Let S_v for each client v denote the tree that v will use to reach all of its commodities. When the algorithm begins, these trees are empty. When the algorithm completes, these trees will define the solution, since each commodity may be built at any intersection of the trees of all interested clients.

- Let S_t^- for each commodity t denote the minimum tree needed to connect the client trees of all clients interested in t. When the algorithm begins, each S_t^- is just the tree connecting all clients interested in t. When the algorithm completes, each of these trees consists of a single node at which the commodity can be built.

We solve this version using Algorithm 2. Each iteration of Algorithm 2 will consider some edge e which is a leaf edge of at least one commodity tree (say it is a leaf for all commodities in set C) and will decide some (possibly empty) set of commodities C_e^* that will definitely be built on the tree side of the edge. We choose these commodities by noting that either all interested clients located on the leaf-side of the edge (we call this set $L(e, C)$ in the algorithm) or all interested clients on the tree-side of the edge ($N(e, C)$) will have to cross the edge. All clients in $L(e, C)$ must have trees that are adjacent to e (by definition of the commodity trees), so if it is cheaper to add e to the trees for clients in $L(e, C)$ than it would be to add e to the trees for clients in $N(e, C)$ (regardless of what other edges might also have to be added for clients in $N(e, C)$), then we can certainly add e for the clients in $L(e, C)$. We show that at all times, at least once leaf edge has a non-empty set C_e^*, so we will chip away at the commodity trees until each tree consists of a single node.

Before analyzing the algorithm, we make a note about line 3. Here, we examine each edge $e = (l, u)$ connecting a leaf of a commodity tree to the commodity tree. Notice that when a commodity tree consists of only a single edge, this edge will be counted twice: once with each node as the leaf. An edge may also be counted twice (once in each direction) if two commodity trees intersect at only this edge.

Figure 3 shows the state of the client and commodity trees at some point during the execution of the algorithm for a small example. The client trees are circled. The dashed line represents a commodity t_1 which is in the interest set of each of v_2, v_3, v_4, v_5 and v_6. The dotted line represents a commodity t_2 which is in the interest set of v_1 and v_3. If we look at edge e, C_e includes both of these commodities. The set C_e^* that minimizes $|L(e, C_e^*)| + |N(e, C_e - C_e^*)|$ includes both t_1 and t_2, so both clients v_2 and v_3 will cross edge e in the final solution.

Algorithm 2 Algorithm for optimal solution to the 1-Median version on a tree.

1: For each client v, initialize tree $S_v = \{v\}$.
2: For each commodity t, initialize tree S_t^- = the minimum tree that connects all clients interested in t.
3: **for** each edge $e = (l, u)$ such that (a) l is a leaf of some commodity tree S_t^- and (b) u is a non-leaf node of S_t^- **do**
4: Define C_e = the set of commodities t such that (a) l is a leaf of S_t^-, (b) there exists some client a with $t \in I(a)$, $l \in S_a$, $u \notin S_a$, and (c) there exists some client b with $t \in I(b)$, $l \notin S_b$ (notice that we do *not* require $u \in S_b$).
5: Define $L(e, C)$ for any subset $C \subseteq C_e$ to be the set of clients a with $l \in S_a$, $u \notin S_a$, a interested in some commodity in C.
6: Define $N(e, C)$ for any subset $C \subseteq C_e$ to be the set of clients b with $l \notin S_b$, b interested in some commodity in C (again, we do *not* require $u \in S_b$).
7: Find the subset $C_e^* \subseteq C_e$ that minimizes $|L(e, C_e^*)| + |N(e, C_e - C_e^*)|$.
8: For all clients $c \in L(e, C_e^*)$, set $S_c = S_c \cup \{u\}$.
9: For each commodity $t \in C_e^*$, set $S_t^- = S_t^- - \{l\}$. If S_t^- = a single node v: t will meet at v, and we are finished with t.

Now we must show the following:

1. Algorithm 2 terminates. For this, it will suffice to show that in any iteration, there is some leaf l (edge $e = (l, u)$) such that $C_e^* \neq \emptyset$ (see Lemma 6.1 below). As long as some C_e^* is non-empty, some commodity tree will shrink (and some corresponding client trees will grow). Since the algorithm terminates when all commodity trees are empty, this leads to termination in less than $m \cdot |E|$ iterations.

2. Algorithm 2 returns a correct solution (each client meets all commodities in its interest set, each commodity meets at a single node). This follows from the definition of the algorithm, since each client tree always intersects each commodity tree for commodities in the client interest set, and from termination, since the algorithm only terminates

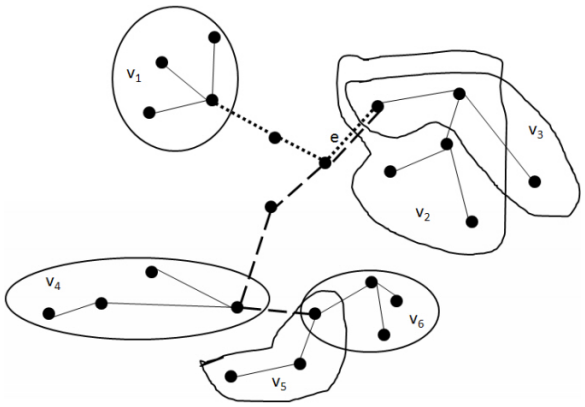

Figure 3: An example of Algorithm 2. This shows the state of the client and commodity trees at some point during the execution of the algorithm for a small example. The client trees are circled. The dashed line represents a commodity t_1 which is in the interest set of each of v_2, v_3, v_4, v_5 and v_6. The dotted line represents a commodity t_2 which is in the interest set of v_1 and v_3. If we look at edge e, C_e includes both of these commodities. The set C_e^* that minimizes $|L(e, C_e^*)| + |N(e, C_e - C_e^*)|$ includes both t_1 and t_2, so both clients v_2 and v_3 will cross edge e in the final solution.

when each commodity meets at a single node.

3. Algorithm 2 returns an optimal solution. We will show in Lemma 6.2 that there exists an optimal solution with the same client trees as those chosen by Algorithm 2.

4. Algorithm 2 can be implemented to run in polynomial time. If the algorithm terminates in less than $m \cdot |E|$ iterations, and each iteration consists of finding C_e^* for each of the (at most) $2|E|$ leaf edges (each edge can be a leaf edge twice per iteration – once in each direction), we only need to show that we can find C_e^* for an edge e in polynomial time. We show this in Lemma 6.3.

LEMMA 6.1. *Algorithm 2 terminates.*

Proof. As stated above, it will suffice to show that in any iteration, there is some leaf l (edge $e = (l, u)$) such that $C_e^* \neq \emptyset$.

First we will show

$$\sum_{e=(l,u)} |N(e, C_e)| \geq \sum_{e=(l,u)} |L(e, C_e)| \tag{6.4}$$

where $e = (l, u)$ is an edge connecting a commodity leaf to the rest of the commodity tree, as defined in Line 3 of Algorithm 2.

Consider how many times an arbitrary client c contributes to each side of Equation (6.4). c will be counted on the right hand side once for each distinct edge $(l_1, u_1), (l_2, u_2), \ldots (l_k, u_k)$ such that $l_i \in S_c$, $u_i \notin S_c$, l_i is a leaf of $S_{t_i}^-$ for some commodity $t_i \in I(c)$, $u_i \in S_{t_i}^-$. We can specify some such commodity t_i for edge (l_i, u_i) causing c to appear once on the right hand side.

For each of these commodities t_i, c will also appear at least once on the left hand side, since each $S_{t_i}^-$ must have at least one other leaf l_i' (connected to $S_{t_i}^-$ by edge e_i', or else we would have removed t_i at Line 9 of Algorithm 2. Since $u_i \notin S_c$, it must also be true that $l_i' \notin S_c$, so c will count as part of $N(e', C_{e'})$.

Suppose we have two of these "other leaves" with $l_i' = l_j'$ $(i \neq j)$. By construction, $(l_i, u_i) \neq (l_j, u_j)$. Also, $l_i' \in S_{t_i}^-$, $u_i \in S_{t_i}^-$, $l_i' = l_j' \in S_{t_j}^-$, $u_j \in S_{t_j}^-$, $(l_i, u_i) \in S_c$, and $(l_j, u_j) \in S_c$. Now, since $S_{t_i}^-$, $S_{t_j}^-$ and S_c are all trees, there exists a cycle from l_j across the edge to u_j, then through $S_{t_j}^-$ to l_j' $(= l_i')$, then through $S_{t_i}^-$ to u_i, along its edge to l_i, then through S_c back to l_j. However, our original graph was a tree, so there cannot be such a cycle. Therefore, each client will appear as many times on the left hand side of Equation (6.4) as on the right hand side, and Equation (6.4) holds.

Because each side of Equation (6.4) contains a sum over all $e = (l, u)$, there must be some such single e with $|N(e, C_e)| \geq |L(e, C_e)|$. For this e, $|L(e, \emptyset)| + |N(e, C_e)| \geq |L(e, C_e)| + |N(e, \emptyset)|$, so C_e^* (the set that minimizes $|L(e, C_e^*)| + |N(e, C_e - C_e^*)|$) is not empty, as desired.

LEMMA 6.2. *There exists an optimal solution with the client trees chosen by Algorithm 2.*

Proof. We will prove that there exists an optimal solution whose client trees are supersets of the client trees chosen by Algorithm 2. Clearly, an optimal solution will not have trees strictly larger than those chosen by Algorithm 2, so this is sufficient to prove the lemma.

The proof is by induction on the iterations of Algorithm 2. At the start of the algorithm, the client trees are empty. Any optimal solution has client trees that are a superset of empty trees. Now, we will assume that after some iteration of the algorithm, there is an optimal solution that includes the partial client trees created so far by the algorithm, and show that the next iteration of the algorithm does not add non-optimal edges to the client trees.

Consider any edge $e = (l, u)$ examined in the next iteration (l is the leaf of a commodity tree S_t^-, $u \in S_t^-$). We have defined set C_e = the set of commodities t such that there exists client a with $t \in I(a)$, $l \in S_a$,

$u \notin S_a$ and there exists client b with $t \in I(b)$, $l \notin S_b$. In any solution, each commodity in C_e must meet on one side or the other of edge e. Suppose there is an optimal solution such that the commodities in $C_{OPT} \subseteq C_e$ meet on the u side of e, and the commodities in $C_e - C_{OPT}$ meet on the l side of e. (In fact, by the inductive assumption and the fact that l is a leaf of each commodity in C_e, all commodities in $C_e - C_{OPT}$ meet exactly at l.) If $C_e^* \subseteq C_{OPT}$, then our new client trees are still a subset of the same optimal solution. So assume $C_e^* - C_{OPT} \neq \emptyset$. We will show that moving the meeting points of the commodities in $C_e^* - C_{OPT}$ from l to u does not increase the cost of the optimal solution.

Before the change to the optimal solution, commodities in C_{OPT} met on the u side of e, those in $C_e - C_{OPT}$ met on the l side. After the change, commodities in C_e^* meet on the u side, and those in $C_e - C_e^*$ meet on the l side. The new cost is equal to

(6.5)
$$\text{optimal cost} + d(e)\Big(|L(e, C_e^*) - L(e, C_{OPT})| - |N(e, C_e - C_{OPT}) - N(e, C_e - C_e^*)|\Big)$$

(6.6)
$$= \text{optimal cost} + d(e)\Big(|L(e, C_e^*)| - |L(e, C_e^*) \cap L(e, C_{OPT})| - |N(e, C_e - C_{OPT})| + |N(e, C_e - C_{OPT}) \cap N(e, C_e - C_e^*)|\Big)$$

(6.7)
$$= \text{optimal cost} + d(e)\Big(|L(e, C_e^*)| - |L(e, C_e^*) \cap L(e, C_{OPT})| - |N(e, C_e - C_{OPT})| + |N(e, C_e - C_e^*)| - |N(e, C_e - C_e^*) - N(e, C_e - C_{OPT})|\Big)$$

(6.8)
$$= \text{optimal cost} + d(e)\Big(|L(e, C_e^*)| + |N(e, C_e - C_e^*)| - |L(e, C_e^*) \cap L(e, C_{OPT})| - |N(e, C_e - C_{OPT})| - |N(e, C_e - C_e^*) - N(e, C_e - C_{OPT})|\Big)$$

(6.9)
$$\leq \text{optimal cost} + d(e)\Big(|L(e, C_e^*)| + |N(e, C_e - C_e^*)| - \big(|L(e, C_e^* \cap C_{OPT})| + |N(e, C_e - (C_{OPT} \cap C_e^*))|\big)\Big)$$

(6.10)
$$\leq \text{optimal cost}$$

Line 6.5 is because we add the cost for clients who have to cross from l to u to reach commodities in C_e^* but didn't have to cross to meet any commodities in C_{OPT}, while we subtract the cost for clients who had to cross from u to l to meet commodities in $C_e - C_{OPT}$ but don't have to cross to meet commodities in $C - C_e^*$.

Line 6.6 is true because the set of clients interested in C_e^* but not interested in anything in C_{OPT} is the same as the set of clients interested in C_e^* minus the set of clients interested both in something in C_e^* and in something in C_{OPT} (similarly with $C - C_e^*$ and $C - C_{OPT}$).

Line 6.7 is true because the set of clients interested in something other than C_e^* *and* in something other than C_{OPT} is the same as the set of all clients interested in something other than C_e^* minus the set of clients interested in something other than C_e^* but not in something other than C_{OPT}.

Line 6.8 is just rearranging terms.

Line 6.9 is true because the set of clients interested in commodities in both C_e^* and C_{OPT} is a subset of the set of clients interested in some commodity in C_e^* and some commodity in C_{OPT}. The set of clients interested in any commodity not in C_e^* minus the set of clients interested in any commodity not in C_{OPT} is the same as the set of clients only interested in commodities in C_{OPT} but not only interested in commodities in $C_{OPT} \cap C_e^*$. Adding this to the set of clients interested in any commodity not in C_{OPT} gives the set of clients interested in any commodity in $C_e - (C_{OPT} \cap C_e^*)$.

Finally, Line 6.10 is true because we chose C_e^* as the set that minimizes $|L(e, C_e^*)| + |N(e, C_e - C_e^*)|$.

LEMMA 6.3. *We can find C_e^* for an edge e in polynomial time.*

Proof. We will solve for an edge $e = (l, u)$ during an iteration by using a directed minimum cut algorithm. Create a directed graph with 5 levels of nodes as follows: Level 1 (source node) = $\{s\}$. Level 2 (tree-side client nodes) = $N(e, C_e)$. Level 3 (commodity nodes) = C_e. Level 4 (leaf-side client nodes) = $L(e, C_e)$. Level 5 (sink node) = $\{t\}$.

Let $M = \max(n, m)$. There is a directed edge of weight 1 from s to each node in Level 2, and a directed edge of weight 1 from each node in Level 4 to t. There is an edge of weight $M + 1$ from each tree-side client node to each commodity node in the client's interest set, and an edge of weight $M + 1$ from each commodity node to each interested leaf-side client node.

The directed min $s-t$ cut of this graph will give the minimum C_e^*. If there is a path from s to a commodity,

it is part of C_e^*. If there is a path from the commodity to t, it is part of $C_e - C_e^*$.

CLAIM 6.1. *A directed min s-t cut of this graph with weight W will give a set C_e^* with $|L(e, C_e^*)| + |N(e, C_e - C_e^*)| = W$*

Proof. In a cut, there will be no paths remaining from s to t. In a min cut, none of the weight $M+1$ edges will be cut (since there are at most M nodes in each level). Therefore, W = the number of cut edges s to Level 2 + the number of cut edges Level 4 to t.

We set C_e^* = the set of all commodities reachable from s. Therefore, the edge from each leaf-side client (interested in a commodity in C_e^*) to t must be part of the cut. So we've cut one edge for each leaf-side client interested in some commodity in C_e^*, which is exactly the client set $L(e, C_e^*)$ (by definition of $L(e, C_e^*)$).

Similarly, if there is a path from a commodity to t (the commodity is in $C_e - C_e^*$), each edge from s to a tree-side client (interested in one of these commodities) must be cut. So we've also cut one edge for each tree-side client interested in a commodity in $C_e - C_e^*$, which is exactly the client set $N(e, C_e - C_e^*)$.

So the number of edges cut (all of weight 1) is exactly $|L(e, C_e^*)| + |N(e, C_e - C_e^*)|$

CLAIM 6.2. *Any C_e^* corresponds to a valid directed cut of this graph with weight = $|L(e, C_e^*)| + |N(e, C_e - C_e^*)|$*

Proof. Given C_e^*, cut each edge that goes from s to a Level 2 node corresponding to a client in $N(e, C_e - C_e^*)$ and cut each Level 4 to t edge corresponding to a client in $L(e, C_e)$. Now, consider any path from s to t. It must pass through the node for some commodity j. We consider two cases. The first case is when $j \in C_e^*$. In this case, all interested leaf-side clients are in the set $L(e, C_e^*)$. Therefore, no matter what edge we follow from the commodity node to a leaf-side client node, the edge from the client node to t has been cut. The second case is when $j \in C_e - C_e^*$. In this case, all interested tree-side clients are in the set $N(e, C_e - C_e^*)$. Therefore, there are no un-cut edges from s to a tree-side client node connected to the commodity. So we have a valid cut.

Since we can find a minimum cut in polynomial time, this concludes the proof of the lemma.

Together, Lemmas 6.1, 6.2, and 6.3 prove the following.

THEOREM 6.1. *Algorithm 2 is a polynomial time algorithm for optimally solving the 1-Median version of* MGSFL.

Although we give an optimal solution on trees, we can show that the 1-Median version is **NP**-hard on general metrics via a reduction from Steiner Tree.

Given a Steiner Tree instance, create an MGSFL 1-Median instance with the same graph. Create one commodity t_v for each node v to be included in the Steiner tree, and 2 clients located at v, each interested in only t_v. Also add one client c located at the root, interested in all commodities. Now, consider the location of each facility in an optimal MGSFL solution. The cost of the solution will include the path from the root to each location as well as twice the path length from each v to the corresponding facility location. This cost will naturally be lower if only client c moves, so all commodities will be built at the Steiner tree locations, and the solution tree for client c will be the minimum Steiner tree.

To approximate the 1-Median version of MGSFL on general metrics, we can use the results from [7] to embed an arbitrary metric into a tree with distortion at most $O(\log n)$. Solving on the resulting tree will give an $O(\log n)$ approximation for the metric.

7 Concluding Remarks

There is a gap between the upper and lower bounds we have established for each version of MGSFL. The major problem left open by our work is whether a polylogarithmic-approximation ratio is achievable in polynomial time for general MGSFL; we have achieved a $2^{O(\sqrt{\log n} \log\log n)}$ approximation, but the best hardness result is an $\Omega(\log^{2-\epsilon} m)$-factor. The same hardness bound applies for the version with laminar interest sets, but our ratio $O(\log^4 n \log m)$ is still quite a bit higher. We give an $O(\log n)$ approximation for general metrics when building costs are uniform across locations, but a constant approximation may be possible. In general, all of our approximations for general metrics incur a cost of $O(\log n)$ factor owing to the reduction to tree metrics; it is unclear whether this gap is inherent. Nevertheless, we have made significant initial progress on this new problem.

There are other variants that merit study. First, just as we considered laminar constraints on client interest sets, the same problem could be defined with "laminar commodities": given any two commodities, the set of clients interested in them do not intersect or one set of clients is entirely contained in the other. We hope to be able to reduce general MGSFL into a small number of instances with laminar clients or laminar commodities. Second, instead of using building costs for commodities, we could place a bound k_t on the number of facilities that need to be built for each commodity t. The resulting problem is a k-median variant of

MGSFL, an extension of the 1-median variant discussed in Section 6.

References

[1] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristic for k-median and facility location problems. *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 21–29, 2001.

[2] R. C. Chakinala, A. Kumarasubramanian, K. A. Laing, R. Manokaran, C. Pandu Rangan, and R. Rajaraman. Playing push vs pull: models and algorithms for disseminating dynamic data in networks. *Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 244–253, 2006.

[3] Moses Charikar, Chandra Chekuri, Ashish Goel, and Sudipto Guha. Rounding via trees: deterministic approximation algorithms for group Steiner trees and k-median. *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 114–123, 1998.

[4] Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k-median problem (extended abstract). *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 1–10, 1999.

[5] Chandra Chekuri, Guy Even, and Guy Kortsarz. A greedy approximation algorithm for the group Steiner problem. *Discrete Applied Mathematics*, 154(1):15–34, 2006.

[6] Fabián Chudak and David B. Shmoys. Improved approximation algorithms for a capacitated facility location problem. *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 875–876, 1999.

[7] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 448–455, 2003.

[8] Naveen Garg, Goran Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group Steiner tree problem. *Journal of Algorithms*, 37(1):66–84, 2000.

[9] Sudipto Guha and Samir Khuller. Greedy strikes back: improved facility location algorithms. *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, pages 649–657, 1998.

[10] Eran Halperin, Guy Kortsarz, Robert Krauthgamer, Aravind Srinivasan, and Nan Wang. Integrality ratio for group Steiner trees and directed Steiner trees. *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 275–284, 2003.

[11] Eran Halperin and Robert Krauthgamer. Polylogarithmic inapproximability. *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 585–594, 2003.

[12] Ara Hayrapetyan, Chaitanya Swamy, and Éva Tardos. Network design for information networks (extended abstract). *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 933 – 942, 2005.

[13] Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and k-Median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM (JACM)*, 48(2):274–296, 2001.

[14] Madhukar R. Korupolu, C. Greg Plaxton, and Rajmohan Rajaraman. Analysis of a local search heuristic for facility location problems. *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, pages 1–10, 1998.

[15] Mohammad Mahdian, Yinyu Ye, and Jiawei Zhang. Improved approximation algorithms for metric facility location problems. *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 229–242, 2002.

[16] Martin Pal, Éva Tardos, and Tom Wexler. Facility location with nonuniform hard capacities. *Proceedings of the 42nd IEEE symposium on Foundations of Computer Science*, pages 329–338, 2001.

[17] R. Ravi and A. Sinha. Multicommodity facility location. *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 342–349, 2004.

[18] G. Reich and P. Widmayer. Beyond Steiner's problem: A VLSI oriented generalization. *Proceedings of the fifteenth international workshop on Graph-theoretic concepts in computer science*, 411:196–210, 1990.

[19] David B. Shmoys, Éva Tardos, and Karen Aardal. Approximation algorithms for facility location problems (extended abstract). *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 265–274, 1997.